Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

R tools user meeting

An-Chi Ho and Núria Pérez-Zanón

contributors: Giulia Carella and Lluís Palma

05/02/2021

Icebreaker

Agenda

1. Icebreaker: A little interaction
2. Package update
   - s2dv
   - startR
   - multiApply
   - CSIndicators
3. DST - S2S4E (Lluís)
4. Gaussian Processes (Giulia)
5. Q&A

Package update

s2dv

- New functions: ACC, PlotACC, EOF, ProjectField, Ano_CrossValid, NAO, PlotBoxWhisker
- We need volunteers to review these new functions 
  (Thanks Danila for reviewing ACC & PlotACC)

Here are some steps for testing a function:

1. Check documentation
2. Find a set of data to test
3. Modify the inputs of the function
4. Compare s2dverification vs s2dv by running:

s2dverification::function_name
s2dv::function_name

s2dv

- Discussion:

  ACC() doesn't have p-value if bootstrapping is used. Do you want to have it? How?

  [parametric method]

```
t <- qt(conf.lev, eno - 2)  # a number
p.val[iexp, iobs] <- sqrt(t^2 / (t^2 + eno - 2))
```

startR

- There is a development version (v2.1.0-2) installed in the machines (not published on CRAN)
  - fixing a small bug that appeared when submitting jobs with Compute() to Nord3. [Solved #84]
  - Force return_vars to have value when inner dim has dependency on file dim. [Solved #88]
- Correct the date/time attributes when the unit of time is ‘month’.
  → Please let us know if you detect any wrong metadata.

startR

- Pay attention when you use *values()* to define the inner dim selector.

  Reminder: If selectors are *values()*, Start() looks for the nearest value in the data. ⟶ It ALWAYS returns something even it is not what you want.

```
* The time value of the request file is "2005-05-16 12:00:00 UTC"
repos_obs <- '/esarchive/obs/ukmo/hadisst_v1.1/monthly_mean/$var$/$var$_$date$.nc'
obs <- Start(dat = repos_obs,
             var = 'tos',
             date = '200505',
             time = as.POSIXct('2005-06-16 12:00:00', tz = 'UTC'),
             latitude = 'all',
             longitude = 'all',
             return_vars = list(latitude = NULL, longitude = NULL, time = NULL),
             retrieve = T)

! Warning: Date selectors have been provided for a dimension defined along a date
!   variable, but no exact match found for all the selectors. Taking the
!   index of the nearest values.
```

startR

- Pay attention when you use *values()* to define the inner dim selector.

  Solution: Use **_tolerance* to ensure the difference is acceptable.

```
* The time value of the request file is "2005-05-16 12:00:00 UTC"
repos_obs <- '/esarchive/obs/ukmo/hadisst_v1.1/monthly_mean/$var$/$var$_$date$.nc'
obs <- Start(dat = repos_obs,
             var = 'tos',
             date = '200505',
             time = as.POSIXct('2005-06-16 12:00:00', tz = 'UTC'),
             time_tolerance = as.difftime(1, units = 'days'),
             latitude = 'all',
             longitude = 'all',
             return_vars = list(latitude = NULL, longitude = NULL, time = NULL),
             retrieve = T)
```

Error in Start(dat = repos_obs, var = "tos", date = "200505", time = as.POSIXct("2005-06-16 12:00:00", :
The selectors of time are out of range [2005-05-16 12:00:00, 2005-05-16 12:00:00].

startR

- Pay attention when you use *values()* to define the inner dim selector.

  Reminder: Defining the dependency in return_vars helps get the correct data and metadata.

```
repos_obs <- '/esarchive/obs/ukmo/hadisst_v1.1/monthly_mean/$var$/$var$_$date$.nc'
time_vector <- as.array(as.POSIXct(c('2005-05-16', '2005-06-16'), tz = 'UTC'))
time_array <- as.array(time_vector)
dim(time_array) <- c(date = 2, time = 1)

obs <- Start(dat = repos_obs,
             var = 'tos',
             date = c('200505', '200506'),
             time = time_vector,
             time_across = 'date',
             latitude = 'all',
             longitude = 'all',
             return_vars = list(time = 'date'),
             retrieve = T)
```

```
obs <- Start(dat = repos_obs,
             var = 'tos',
             date = c('200505', '200506'),
             time = time_array,
             latitude = 'all',
             longitude = 'all',
             return_vars = list(time = 'date'),
             retrieve = T)
```

multiApply

- multiApply v2.1.3 has been published on CRAN and installed in the machines

  It is published under **Apache license 2.0**

CSIndicators

- **Climate Services Indicators** is a new package that borns from the need of ESS group of sharing functions with partners.

  This has some implications like:
  - licence
  - adequated package name and structure

Even though, ClimProjDiags is computing some indicators, it cannot be completely adapted to those requirements because it is being use in ESMValTool.

Therefore, there will be two tools that are able to compute indicators:

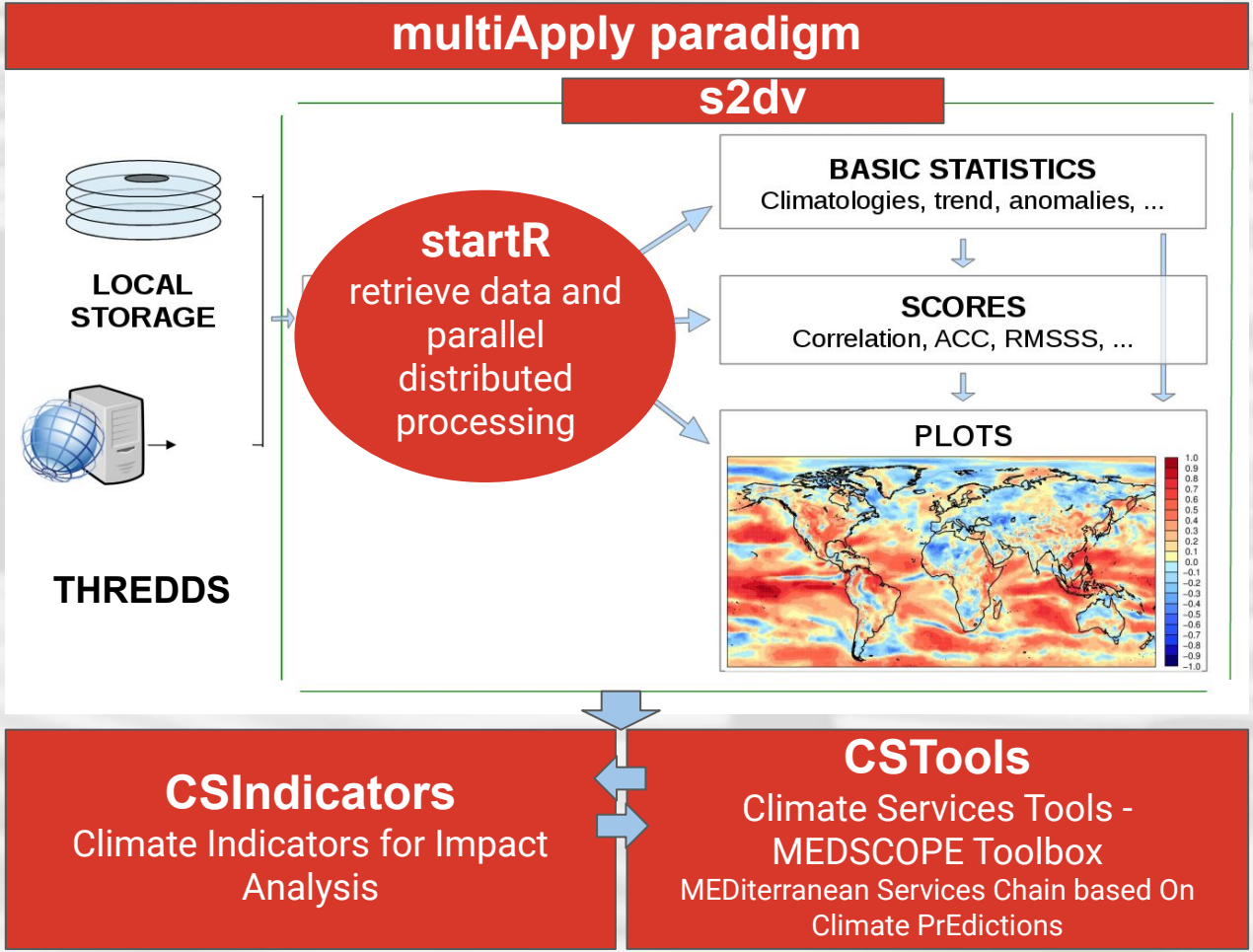
★ ClimProjDiags if you don't need to share functions with partners and you want use the indicators definition in climdex.pcic package (ETCCDI)

★ CSIndicators to share functions with partners and follow ESS group expertise.

Department R Tools scheme

GitLab repo CSIndicators:

https://earth.bsc.es/gitlab/es/csindicators



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

CSIndicators

	Function	Indicators
1	PeriodAccumulation	SprR, HarR, PRCPTOT
2	PeriodMean	GST, SprTX, DTR
3	PeriodMax	TXx, TNx, Rx1day
4	PeriodMin	TXn, TNn
5	PeriodRatio	SDII (Simple precipitation intensity index)
6	PeriodSPEI	SPEI6, SPEI12, ...
7	AccumulationExceedingThreshold	GDD, R95pTOT, R99pTOT
8	TotalTimeExceedingThreshold	SU35, SU36, SU40, SU, FD, ID, TR, R10mm, R20mm, Rnmm
9	PercentageTimeExceedingThreshold	TX90p, TX10p, TN90p, TX10p
10	TotalSpellTimeExceedingThreshold	WSDI, CSDI (Cold spell duration index)
11	MaxSpellTimeExceedingThreshold	CDD (Max Length of Dry Spell), CWD (wet)
12	MaxSpellAccumulation	Rx5day (maximum consecutive 5-day precipitation)
13	SpellIntensityRatioExceedingThreshold	Heat Magnitude: HDM3, ….
14	TimeBetweenSpells	GSL (Growing season length)
15	CapacityFactor	
16	KineticEnergy	
17	Threshold	-
18	QThreshold	-
19	AbsToProbs	-
20	SelectPeriodOnData	-
21	SelectPeriodOnDates	-

Functions to be included in the first version of the package

- One function can be used to compute more than one sectorial indicator:
  - The names indicate which will be the output given the calculation that the function does
  - Explanations about how to get specific indicators will be provided in the documentation
  - The plan is test them for sub-seasonal, seasonal, decadal forecast and also for projections and time-series. Your collaboration will be needed!
- The first functions selected correspond to the ones used in MedGOLD project. For now, tests for Seasonal forecast are the priority.
- Auxiliary functions **SelectPeriodOnDates** and **SelectPeriodOnData** will require exhaustive testing, but we think they would be extremely useful even for other purpose different than indicators computation.

CSIndicators

IndexExceedingThershold **<- function(*data, dates, start,* end,**
***time_dim, threshold, op = '<', spell.length,...*) {**

1. Checks parameters

2. Select period if requested

3. Compute:
res <- **Apply**(list(data), target_dims = ftime_dim, fun = **Atomic**,
…, na.rm = na.rm, ncores = ncores)$output1

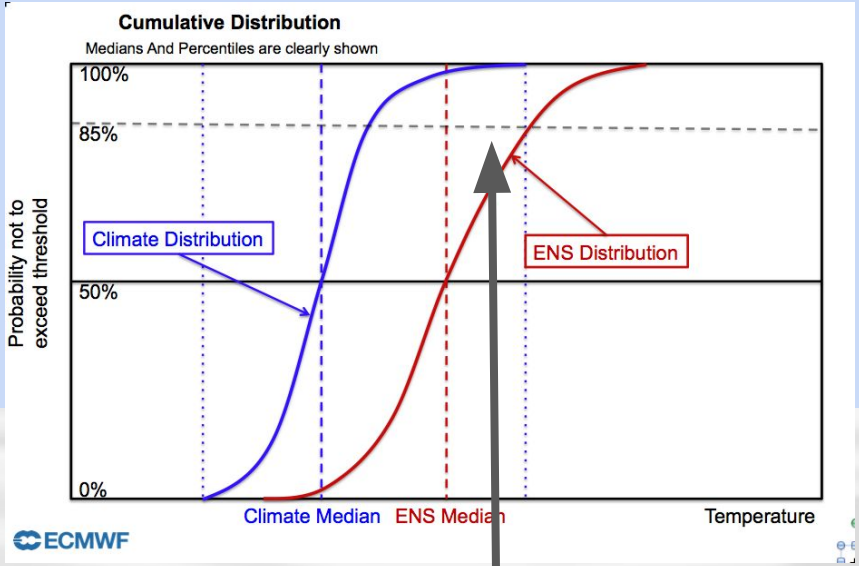
SelectPeriodOnDates()
SelectPeriodOnData()

```
Atomic <- function(x, y, …) {
    sum(x > y)
}
```

Threshold()

1) **Threshold()** uses a reference dataset to obtain the corresponding values of a specific percentile. It returns a grid of thresholds for each julian day. It is used in WSDI.

Reference {time, lat, lon...}, threshold = 85th percentile → result {... lat, lon}



X

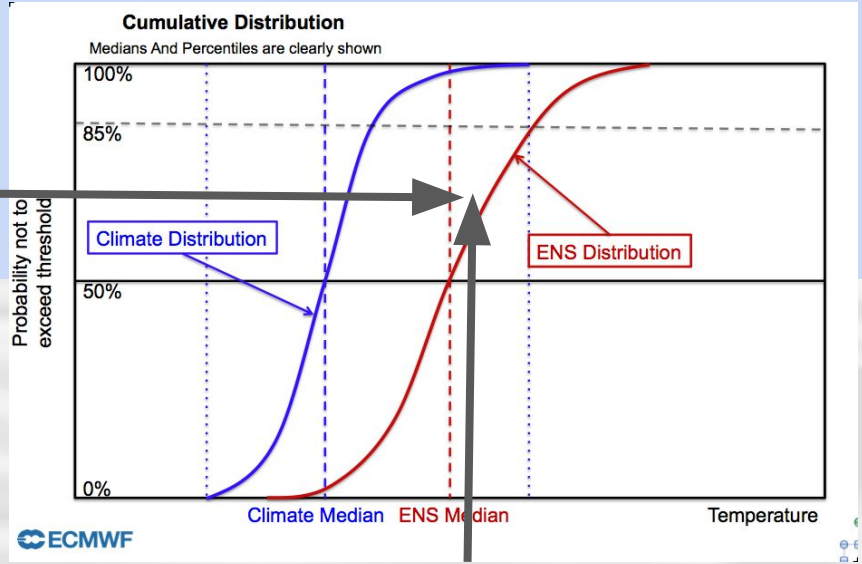
QThreshold()

2) The indices defined with **absolute threshold** can be reinterpreted using a reference:

a) On **observations** (no members): for each grid point on the dataset {time, lat, lon}, the cumulative distribution function is used to calculate which value corresponds to the fix absolute threshold

Relative_threshold[s, ftime, i, j...] = ECDF(data[-s, ftime, i, j]...)(absolute_threshold)

Corresponding Probability = relative threshold



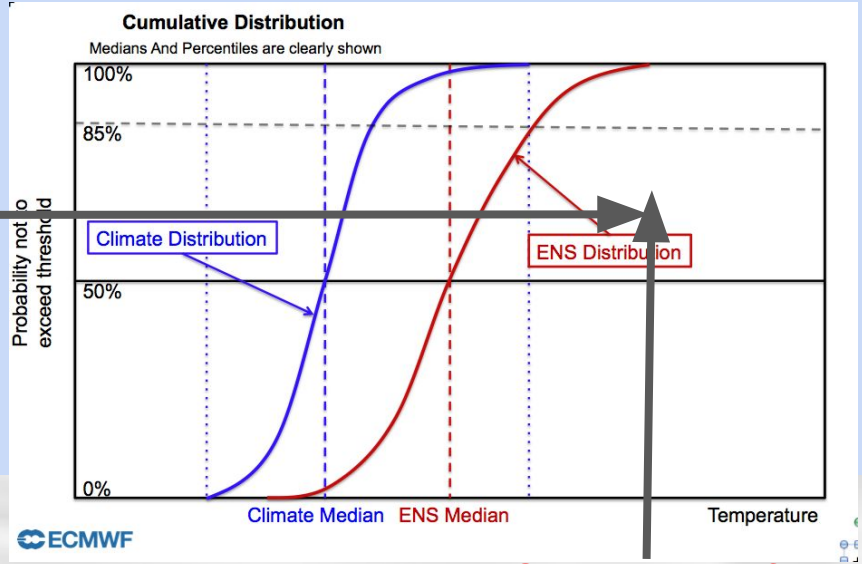
Absolute threshold

AbsToProbs()

3) Once we have the relative threshold, we would like to compare it to the experiment. To do this, we need to transform the experiment to its probabilities:

Exp_prob[m, s, ft, lat, lon...] = ECDF(exp[**ALL**, **-s**, ft, lat, lon...])(exp[m, s, ft, lat, lon...])

Corresponding Probability

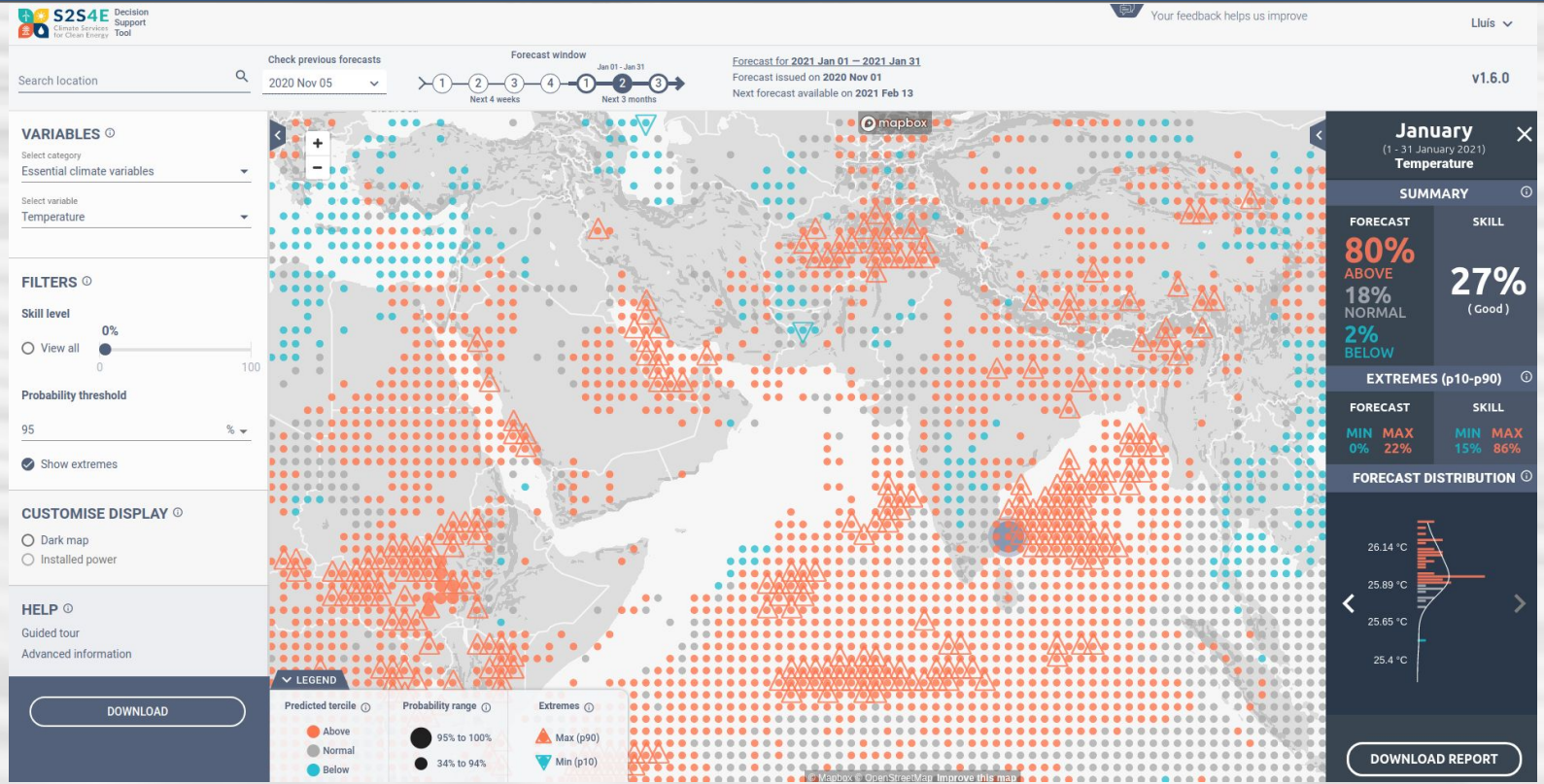


Single value of the

- Exp_prob and relative_trheshold can be compared.

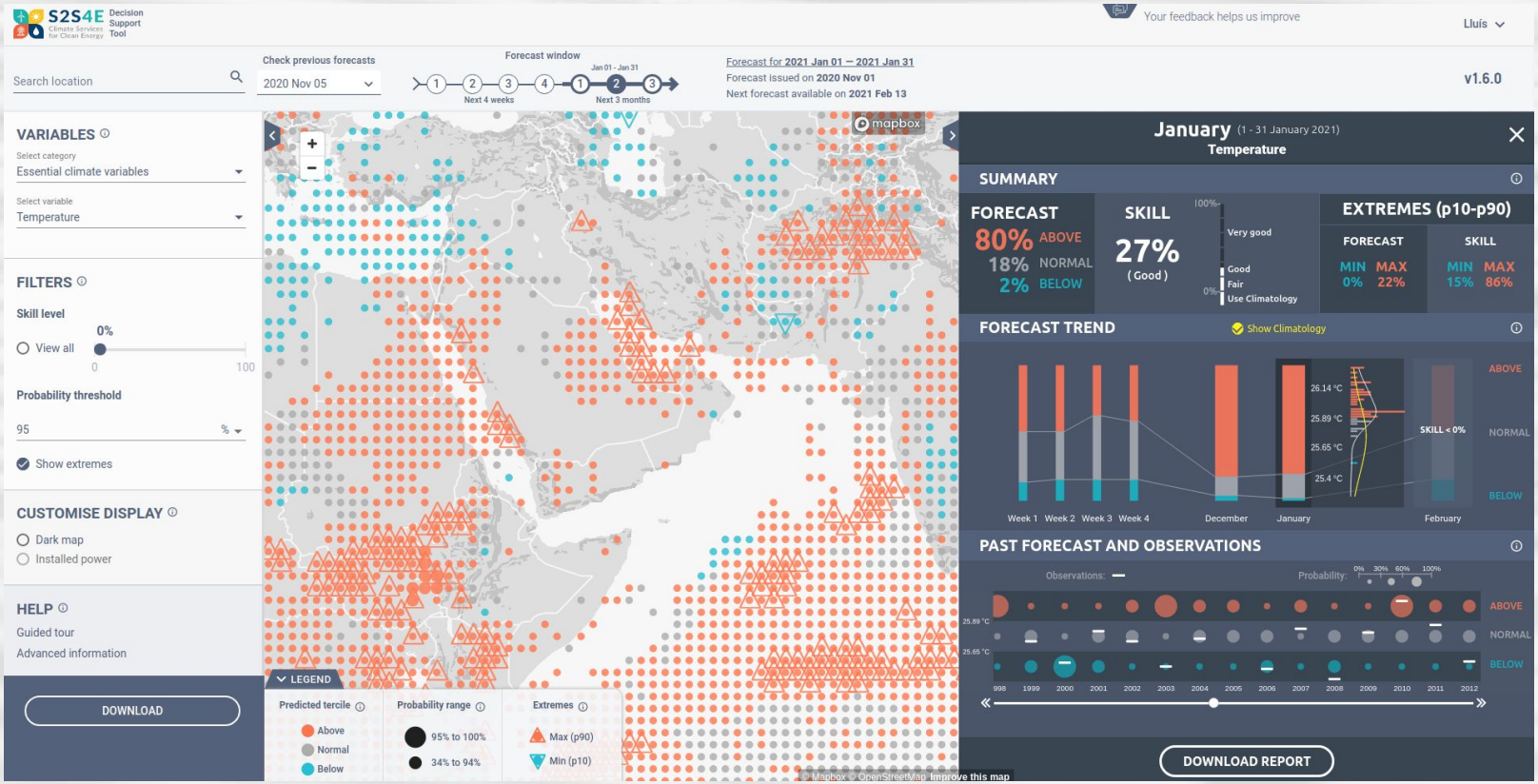
DST S2S4E

DST



https://s2s4e-dst.bsc.es/

DST



https://s2s4e-dst.bsc.es/

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

DST

Similar services but with differences in the details:

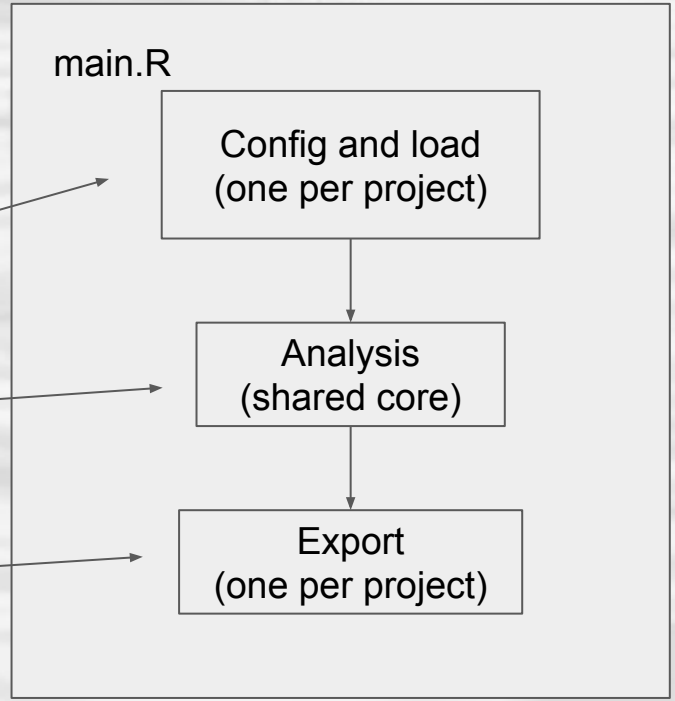
- Systems and variables
- Forecast time & aggregation
- Spatial distribution
- Time frequency of the data used.
- Outcome

- **S2S4E:**
  - Seasonal (monthly) and subseasonal (weekly)
  - 7 ECVs (global + country aggregated)
  - 2 Hydrological variables
  - 5 energy indicators
  - Seasonal multi-model (Temp, sfcWnd)
  - .nc + DST
- **Visca:**
  - Seasonal (monthly and seasonal)
  - 2 ECVs (global and demosites)
  - .json + Platform
- **Decathlon:**
  - Seasonal (monthly + seasonal), subseasonal (weekly)
  - 1 ECV (Spain grid + agg. regions)
  - .nc + PDF Outlook
- **Vitigeoos, Med-gold, Focus...**

DST

Although some similarities could be identified:

- Loading hcst, fcst and obs data and interpolate
- Calibrate using obs data
- Compute probabilistic information and skill metric
- Export the results



https://earth.bsc.es/gitlab/es/S2S4E-backend-BSC/

DST

.conf

```
library(startR)
library(easyNCDF)

library(parallel)
library(pryr) # To check mem usage.

source('export_2_nc.R')
source('S2S/s2s.filefmt.R')

source("s2s_tools.R")
source("Calibration_fcst3.R")
source("R_Reorder.R")
source("R_CST_MergeDims.R")

#-----------------------------------------------------------------------
# Settings
#-----------------------------------------------------------------------

# workflow file
wf <- "s2s.analysis.R"
load.conf <- "S2S/s2s.load.R"

## TODO add as input?
mask.path <- paste0(s2s4e.dir,'/data-analysis/masks/mask_europe_S2S_ecmwf.Rdata')

fcst.type <- 'subseasonal'
system.name <- "s2s-ecmwf"
mm=F
fcst.dir <- "/esarchive/exp/ecmwf/s2s-monthly_ensfor/"
hcst.dir <- "/esarchive/exp/ecmwf/s2s-monthly_ensforhc/"
obs.dir <- "/esarchive/recon/ecmwf/era5/"

remap.method <- "ycon"
Sys.setlocale("LC_TIME", "en_US")

# decreasing sort
```

DST

.load

```
obs <- Start(dat = obs.path,
             var = variable,
                 file_date = dates_file,
                 latitude = 'all',
                 longitude = 'all',
             synonims = list(latitude=c('lat','latitude'),
                             longitude=c('lon','longitude')),
                 return_vars = list(latitude = 'dat',
                                    longitude = 'dat',
                                    time = 'file_date'),
                 split_multiselected_dims = TRUE,
                 retrieve = TRUE)

obs.NA_dates.ind <- Apply(obs,
                          fun=(function(x){ all(is.na(x))}),
                          target_dims=c('time', 'latitude', 'longitude'))[[1]]
obs.NA_dates <- dates_file[obs.NA_dates.ind]
obs.NA_dates <- obs.NA_dates[order(obs.NA_dates)]
obs.NA_files <- paste0(obs.dir,fcst.freq,"/",variable,"_",
                       freq.obs,obs.grid,"/",variable,"_",obs.NA_dates,".nc")

if (any(is.na(hcst))){
  print(" ERROR: MISSING HCST VALUES FOUND DURING LOADING # " )
  print(" ############################################### " )
  print(" ######           MISSING FILES              #### " )
  print(" ############################################### " )
  print("hcst files:")
  print(hcst.NA_files)
  print(" ############################################### " )
  print(" ############################################### " )
  quit(status=1)
}
```

DST

```
obs <- Start(dat = obs.path,
             var = variable,
                 file_date = dates_file,
                 latitude = 'all',
                 longitude = 'all',
             synonims = list(latitude=c('lat','latitude'),
```

The outcome is an array containing the essential dims for the analysis:

- dat: for different systems (multi-model)
- var: for different variables (indicators)
- member: ensemble dimension (probs)
- syear: Initialization year (verification and calibration)
- sday: Initialization window (Subseasonal calibration)

.load

```
  print(" ############################################## " )
  print("hcst files:")
  print(hcst.NA_files)
  print(" ############################################## " )
  print(" ############################################## " )
  quit(status=1)
}
```

DST

.analysis

```
calibrated_fcst <-Apply(data=list(obs=var.obs,
                                  hcst=var.hcst,
                                  fcst=var.fcst),
                                  extra_info=list(na.rm=na.rm),
                                  target_dims=c('sday','syear','member'),
                                  output_dims=c('member'),
                                  na.rm=na.rm,
                                  ncores=ncores,
                                  fun = .fcstcal)[[1]]

calibrated_fcst[!is.finite(calibrated_fcst)] <- NA

if(mm){
  calibrated_fcst <- MergeDims(calibrated_fcst,
                       merge_dims=c('dat','member'),
                       rename_dim='member')
  calibrated_fcst <- drop_na_dim(calibrated_fcst,'member')
}
```

```
probs.outfile <- get_filename(dir, variable, export.sdate, fcst.sdate,
                              agg, fcst.type, "prob")

fcst.outfile <- get_filename(dir, variable, export.sdate, fcst.sdate,
                              agg, fcst.type, "ensemble")


save_probs(variable, probs.2.export, export.sdate, probs.outfile,
           export.dates, grid, agg, fcst.type)

export.fcst <- Subset(calibrated_fcst, 'sweek', list(1), drop='selected')
save_forecast(variable, export.fcst, export.sdate, fcst.outfile,
              export.dates, grid, agg, fcst.type)

print(paste0("######### ",agg,
             ": FCST PROBS/ENSEMBLES exported to NC ##############"))
```

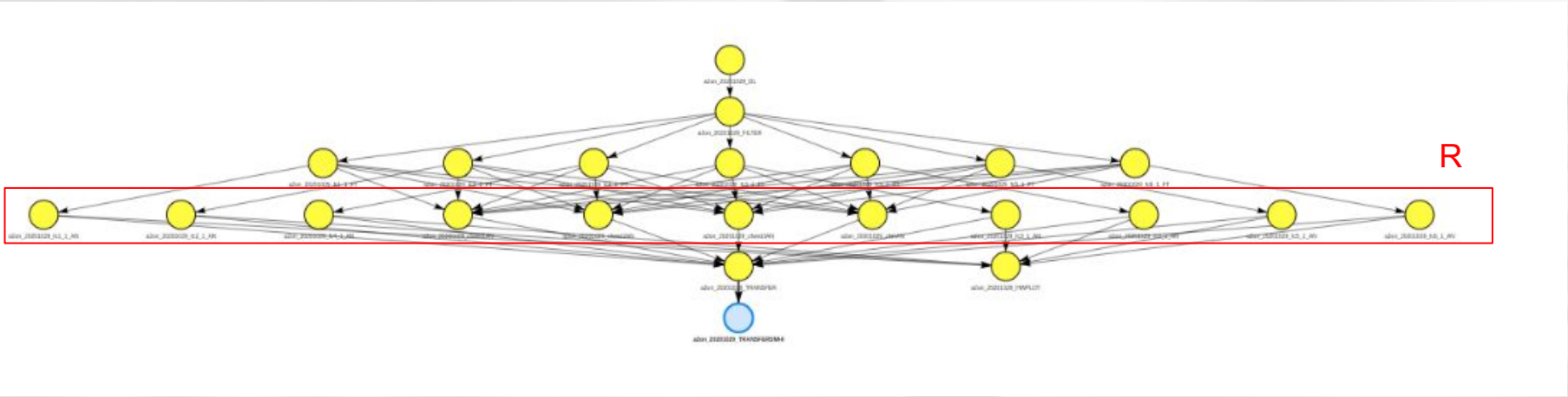
DST

.export

```
save_probs <- function(variable,
                       probs,
                       fcst.sdate,
                       outfile,
                       monthnames,
                       grid,
                       agg,
                       fcst.type) {

   ifelse(exists("lonlat_dctln"),
       lalo <- c('longitude','latitude'), #decathlon subseasonal
       lalo <- c('latitude','longitude')) #no decathlon

  if (tolower(agg) == "global"){
    probs <-  lapply(probs, function(x){
                        Reorder(x, c('bin',lalo, 'time'))})
  }

  pbn <- Subset(probs$tercile, 'bin', list(1), drop='selected')
  pn <-  Subset(probs$tercile, 'bin', list(2), drop='selected')
  pan <- Subset(probs$tercile, 'bin', list(3), drop='selected')
  p10 <- Subset(probs$extreme, 'bin', list(1), drop='selected')
  p90 <- Subset(probs$extreme, 'bin', list(3), drop='selected')

  pn.sdname <- paste('Probability below normal category ', sep='');
  pan.sdname <- paste('Probability above normal category ', sep='');
  pbn.sdname <- paste('Probability normal category ', sep='');
  p10.sdname <- paste('Probability below extreme category ', sep='');
  p90.sdname <- paste('Probability above extreme category  ', sep='');

  if (tolower(agg) == "country"){
    dims <- c('Country', 'time')
    pn.sdanme <- paste0('Country-Aggregated ', pn.sdname)
    pbn.sdanme <- paste0('Country-Aggregated ', pbn.sdname)
    pan.sdanme <- paste0('Country-Aggregated ', pan.sdname)
```

DST

General wf in autosubmit



Gaussian Processes

What is a GP?

Given the observational model: $y = N(f(x), \sigma)$

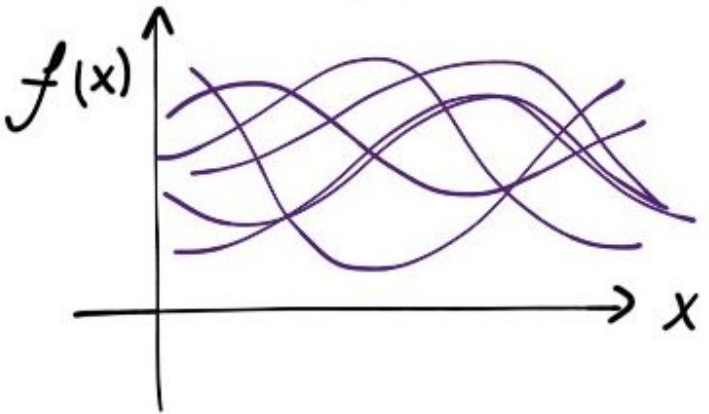
How do we capture complex functional behaviours?

- Polynomials → difficult to regularize their flexibility
- A GP defines probability distributions over function spaces directly
  - Infinite dimensional
  - Parametrized by

$$m : X \longrightarrow \mathbb{R}$$

$$k : X \times X \longrightarrow \mathbb{R}^+$$

$$\pi(f) = GP(m, k)$$



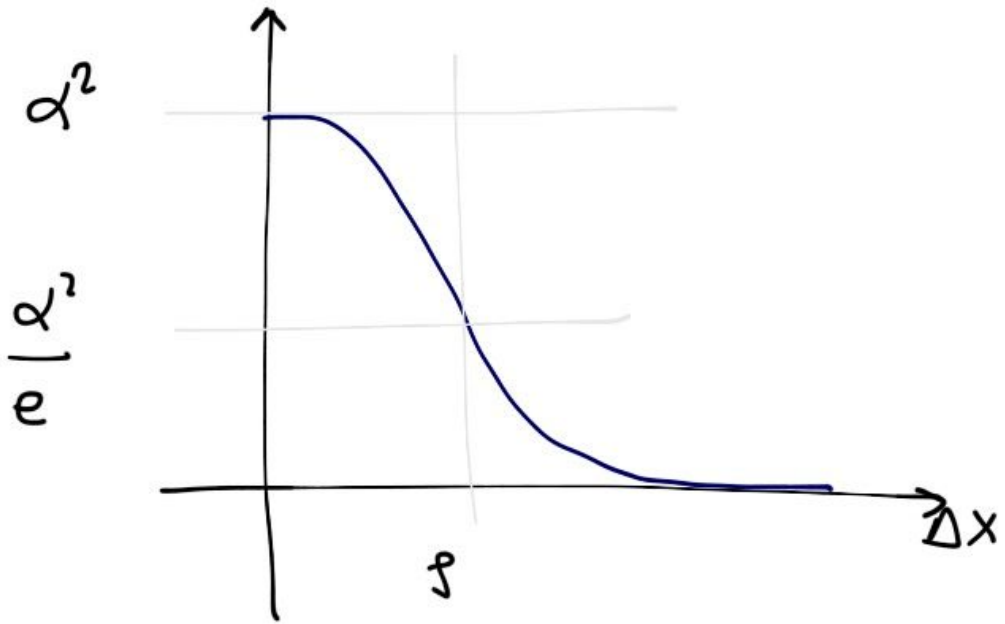
Covariance functions

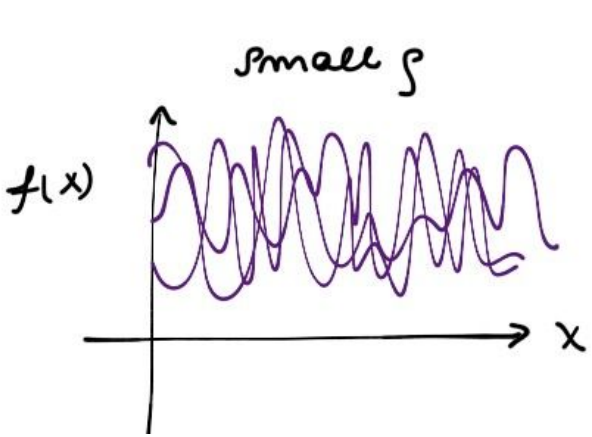
- Zero-mean GPs
- Covariance function:
  - Diagonal elements

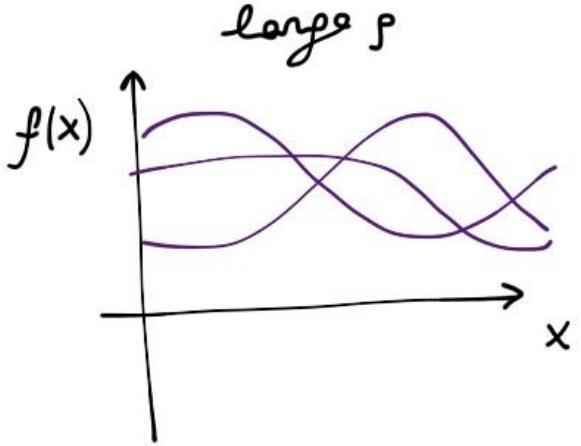
    → marginal variations of function values
  - Off-diagonal elements

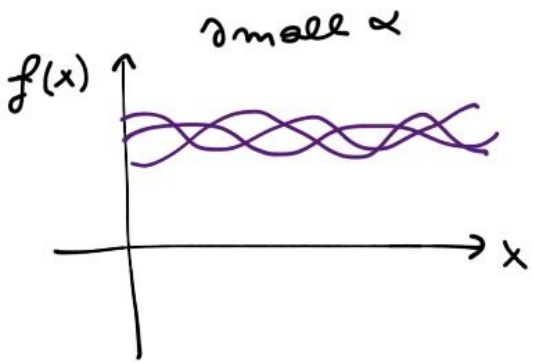
    → correlations between function values
- Typically assumed to be stationary and isotropic → $\Delta x = |x_1 - x_2|$
- Different choices of covariance functions: exponential, spherical, Matérn

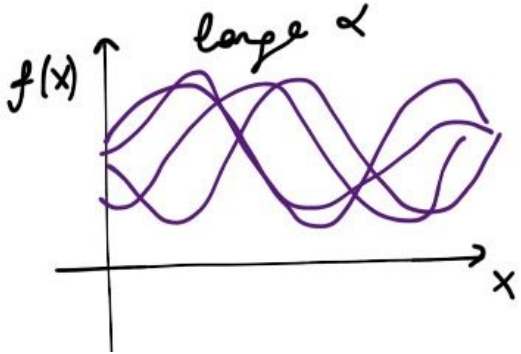
$$k(\Delta x) = \alpha^2 \exp\left(-\frac{1}{2}\left(\frac{\Delta x}{\rho}\right)^2\right)$$




small ρ
f(x)
x

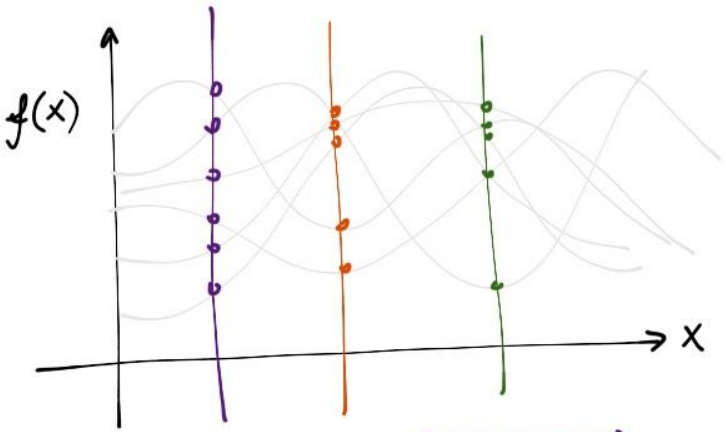

large ρ
f(x)
x


small α
f(x)
x


large α
f(x)
x

GPs in practice

- We'll never be able to manipulate an entire sampled function
- Consider only the function values at a finite number of covariate values (*grid*)
- The function values over the grid follow a distribution specified by multivariate normal



$\pi(f) = GP(m, K) \longrightarrow \pi(f) = N(\underline{m}, \underline{K})$

function → $m_n = m(x_n)$

matrix → $K_{nm} = K(x_n, x_m)$

GPs in practice

- We can also generate predictions

$$m = \begin{pmatrix} m(x_1^{obs}) \\ \vdots \\ m(x_n^{obs}) \\ m(x_1^{pred}) \\ \vdots \\ m(x_m^{pred}) \end{pmatrix}$$

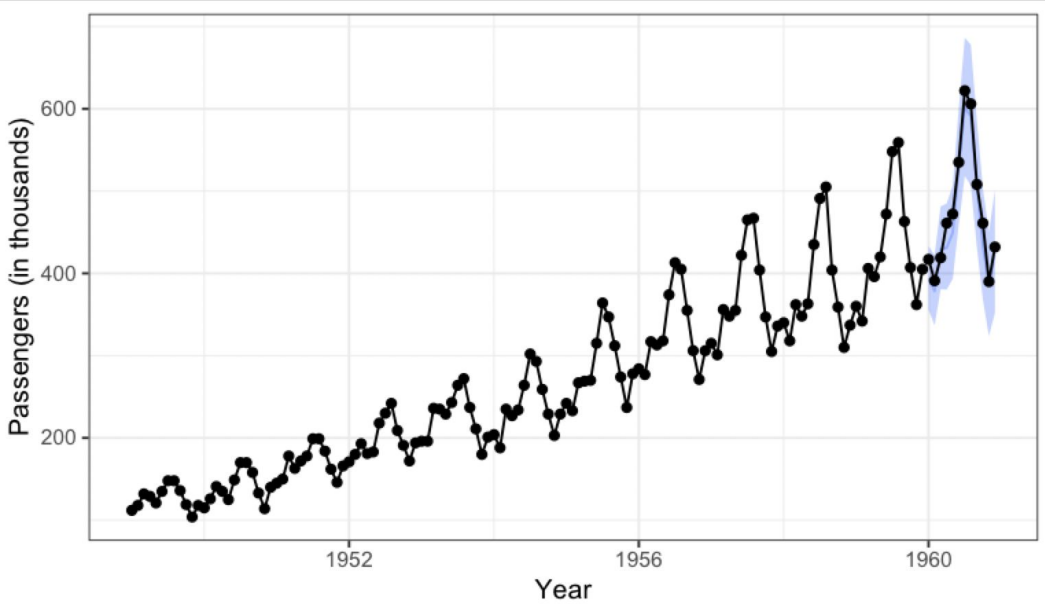
$$K = \begin{pmatrix} k(x_1^{obs}, x_n^{obs}) & \cdots & \cdots \\ \vdots & \ddots & \\ k(x_m^{pred}, x_1^{obs}) & & k(x_m^{pred}, x_m^{pred}) \end{pmatrix}$$

- Inference: Maximum likelihood, Bayesian framework

- The curse of dimensionality: we need to compute the determinant and the inverse of the covariance matrix which scales as $\mathcal{O}(N^3)$ → sparse methods, basis function approximations

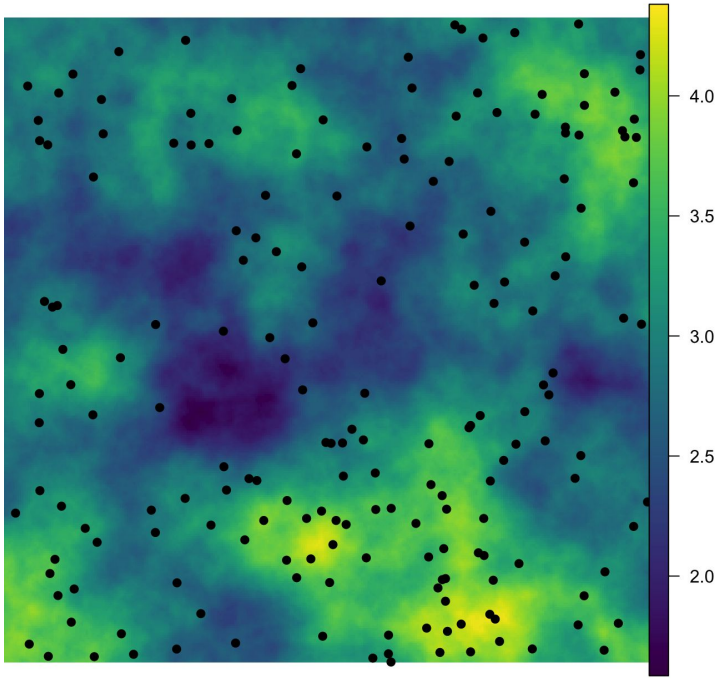
Applications

Time series modeling
(1D interpolation)



https://earth.bsc.es/gitlab/gcarella/gaussian-process-example-in-r

Spatial modelling
(2D interpolation)



R packages to fit GP models

- **RStan**: R interface to the Stan programming language (C++ backend) for fitting Bayesian hierarchical models (MCMC)
  - Very flexible ✅
  - Existing GP approximations for large datasets still too slow for large data and models
- **INLA**: R package (C++ backend) for Bayesian inference for Latent Gaussian Models (LGM)
  - Work well with large-hish data and models ✅
  - Spatio-temporal models for geostatistical data ✅
  - Limited to LGM and the Matèrn class of covariance functions
- **gplite**: R package
  - Easy syntax ✅
  - Simple models only

R packages to fit GP models

- **gstat**: R package for spatio-temporal kriging
  - Ignore the uncertainty in the GP parameters (variogram)
  - Inefficient for large data
- **FKR**: R package for spatio-temporal kriging with large data
  - Ignore the uncertainty in the GP parameters (variogram)
  - Non-stationary covariance ✅

Other references

http://www.gaussianprocess.org/gpml/chapters/

https://becarioprecario.bitbucket.io/inla-gitbook/ch-spatial.html

https://mc-stan.org/docs/2_19/stan-users-guide/gaussian-process-regression.html

Q & A

Next meeting: 5th Mar. 2021 (Friday 4pm)